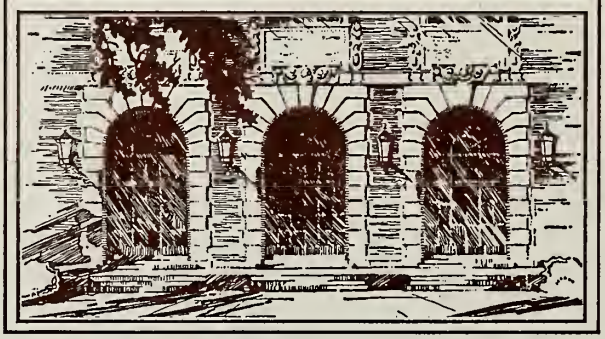

 UIUCDCS-R-74-651

BOUNDS FOR SELECTION

by

Laurent Hyafil

June 1974

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-74-651

Bounds for Selection

by

Laurent Hyafil[†]

June 1974

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by NSF Grant GJ-41538 and in part by IRIA.

[†]IRIA-Laboria, Domaine de Voluceau, 78150 Rocquencourt, FRANCE.

ABSTRACT

In this paper we show that the minimum number of comparisons necessary for the computation of the k^{th} element of a totally ordered set of size $n, V_k(n)$, is lower bounded by $n-k+(k-1)\lceil \log_2(\frac{n}{k-1})\rceil$. For $3 < k < \frac{n}{4}$, this bound improves the best lower bound presently known. A new algorithm which yields an upper bound that is better than the currently known bound for a large range of values of n will also be presented.

1. Introduction

The selection problem is to determine the k^{th} element of a totally ordered set P of size n. Two efficient algorithms for solving this problem are presently known. When k is small with respect to n, $3 < k < \frac{n}{4} - 1$, A. Hadian and M. Sobel's algorithm ([3]) which needs at most $n-k+(k-1)\lceil \ell g(n-k+2) \rceil$ [†] comparisons is adequate. Another method, using at most 5.73 n comparisons, was discovered by M. Blum et al. ([1]). This method is more efficient than Hadian and Sobel's method for $k \geq 4n/\ell g n$.

Let $V_k(n)$ denote the minimum number of comparisons necessary for finding the k^{th} element of a set of size n. The exact values of $V_k(n)$ are known for k = 1 ($V_1(n) = n-1$), and k = 2 ($V_2(n) = n-2+\lceil \ell g n \rceil$) (Schreier and Kislitsyn). For k = 3, F. Yao ([6]) has obtained a lower bound which is equal to the upper bound of Hadian and Sobel for infinitely many values of n. V. Pratt and F. Yao ([5]) also showed that:

for $k \leq \ell g n/2\ell g\ell g n$, $V_k(n) \geq n-k+(k-1)[\ell g n-(k-1)\ell g * n - 2\ell g((k-1)!)]$,

for $k \leq \frac{n}{3}$, $V_k(n) \geq n+2k-\ell g n$,

for $n/3 < k < \lfloor n-3/2 \rfloor$, $V_k(n) \geq (3n+k)/2-\ell g n-O(1)$.

improving the bound due to Blum et al, except when $\ell g n/2\ell g\ell g n < k < \ell g n$.

In this paper we first present a new lower bound for $V_k(n)$, namely: $n-k+(k-1)\lceil \ell g(\frac{n}{k-1}) \rceil \leq V_k(n)$. When $3 < k < \frac{n}{4}$, this bound is strictly greater than the best previously known bound. For instance, for k = 5, this result together with the best known upper bound enables us to determine the value of $V_5(n)$ within a gap of at most 8, while the previously known bounds leave a gap of at least 80. Furthermore, this

---

[†] ℓg stands for $\log_2$.

result shows that, for a fixed value of k, the tree selection algorithm is asymptotically optimal.

We then present a new algorithm for selecting the k^{th} largest element of a set of size n which yields an upper bound that improves strictly the previously known bound when $2\lg n < k < 4n/\lg n$ and when $2^i+k-2 < n \leq 2^{\left\lceil \frac{k-1}{k-2} i \right\rceil -1}$ for all integer i. Specifically, for $k = 3$, the new upper bound is $V_3(n) \leq n-3+\lceil \lg(n-1) \rceil + \lceil \lg(n-2^{\left\lfloor \frac{\lg n}{2} \right\rfloor}) \rceil$.

Following the original formulation of the selection problem by Lewis Caroll ([2]), we call an element of the set P a "player" and a comparison between two players a "match" which must be won by one of the two players. A procedure for selecting the k^{th} largest element will be referred to as a "tournament" for determining the k^{th} best player.

2. The Lower Bound

We want to show that, for any algorithm that computes the k^{th} best player among n players, there exists a ranking of the players such that this algorithm must perform at least $n-k+(k-1)\lceil \lg \frac{n}{k-1} \rceil$ comparisons. The idea of using an oracle in our proof is due to Knuth ([4]) who gave a new proof of Kislitsyn's lower bound for $k = 2$. Here we extend this idea for all values of k. An oracle is basically a deterministic process which builds up a ranking among the players, while the algorithm tries to find out the solution. This ranking, which must satisfy transitivity and antisymmetry, will force the algorithm to perform at least $V_k(n)$ comparisons. A correct algorithm cannot stop before the k^{th} player is uniquely determined by the oracle (how can the algorithm know the answer if the oracle still has some freedom to choose it?). As a direct

consequence, the set of the k-1 best players must also be uniquely determined.

We describe the oracle O as an automaton whose states are represented by ordered pairs. To be specific, the state vector S_t before the t^{th} match is (φ_t, E_t) where φ_t is a mapping from P to N, and E_t is a totally ordered subset of P. The initial state is $S_1 = (I, \emptyset)$ where I is the constant mapping such that $\forall x \in P, I(x) = 1$. Roughly speaking, the players in E_t are the top players, specifically the i^{th} player to enter the set E_t is the i^{th} best player. Candidates for entering E_t are selected according to the values of φ_t.

The input to the oracle at time t is an unordered pair of players $\{x,y\}$, who are engaged in the t^{th} match according to the selection procedure. The oracle decides the winner of the match and enters state S_{t+1} according to the following rules:

R1 - If $x \in E_t$ and $y \in E_t$, then x wins if and only if $x > y$ (E_t is an ordered set). Moreover, $S_{t+1} := S_t$.

R2 - If $x \in E_t$ and $y \notin E_t$, then x wins and $S_{t+1} := S_t$.

R3 - If $x \notin E_t$ and $y \notin E_t$, then if $\varphi_t(x) > \varphi_t(y)$ x wins; and if $\varphi_t(x) = \varphi_t(y)$ an arbitrary decision compatible with transitivity will be made. In both cases, if $\varphi_t(x) + \varphi_t(y) \geq \frac{n}{k-1}$ then $\varphi_{t+1} := \varphi_t$, $E_{t+1} := E_t \cup \{x\}$ and x becomes the smallest element of E_{t+1}. If $\varphi_t(x) + \varphi_t(y) < \frac{n}{k-1}$ then $E_{t+1} := E_t$, $\varphi_{t+1}(y) := o$, $\varphi_{t+1}(x) := \varphi_t(x) + \varphi_t(y)$ and $\forall z \neq x,y \ \varphi_{t+1}(z) := \varphi_t(z)$.

Being given that x dominates only x at time 1, we say that x dominates y at time t+1, if x dominates y at time t, or if x has beaten y in the t^{th} match, or if x dominates z and z dominates y. Clearly, if x dominates y, x is a better player than y.

Theorem: the number $V_k(n)$ satisfies: $n-k+(k-1)\lceil \ell g(\frac{n}{k-1})\rceil \leq V_k(n)$.

We first prove the following lemma:

Lemma: Using oracle 0 the k-1 best players will have played at least $(k-1)\lceil \ell g(\frac{n}{k-1})\rceil$ matches when the tournament is completed.

Proof: The lemma follows from the facts listed below:

Fact 1: The number of matches won by x by time t is greater or equal to $\lceil \ell g \varphi_t(x)\rceil$.

Fact 2: Let $e_i \in E_t$ be the i^{th} player $(1 \leq i \leq |E_t|)$ to enter E_t. Then e_i can be dominated only by e_j with $j \leq i$.

Fact 3: $\sum_{x \in P} \varphi_t(x) = n$.

We call W_t the set of players x such that $x \notin E_t$ and $\varphi_t(x) > o$.

Fact 4: $|E_t| + |W_t| > k-1$.

This is a consequence of Fact 3 and from the fact that: $\forall x \in P\ \varphi_t(x) < \frac{n}{k-1}$.

Fact 5: At the end of the tournament $|E_t| \geq k-1$.

Since the players in W_t can be dominated only by the players in E_t, if $|E_t| < k-1$, then any player in E_t or W_t can be one of the k-1 best players. Contradiction results from Fact 4.

Fact 6: At the end of the tournament the k-1 best players are the k-1 top players in E_t.

This is a consequence of Facts 2 and 5.

Since, when x enters E_{t+1} by defeating y, $\varphi_t(x) + \varphi_t(y) \geq \frac{n}{k-1}$ and $\varphi_t(x) \geq \varphi_t(y)$, the result is a direct consequence of Facts 1 and 6. □

Proof of theorem: According to the lemma, the k-1 best players have played at least $(k-1)\lceil \ell g(\frac{n}{k-1}) \rceil$ matches. Clearly, any player who is not among the k best players have lost at least one match against a player which is not among the k-1 best. Thus, there are n-k additional matches which were not included in the count of the matches played by the k-1 top players. This completes the proof of the theorem.

3. Improving the Upper Bound

Since Hadian and Sobel's algorithm needs at most $n-k+(k-1)\lceil \ell g(n-k+2) \rceil$ comparisons, the new lower bound presented above enables us to determine $V_k(n)$ to within a gap of at most $(k-1)\lceil \ell g(k-1) \rceil$ comparisons. The new algorithm we present reduces that gap when $\ell g\, n < k/2$ and when

$$2^i+k-2 < n \leq 2^i + 2^{\left\lceil \frac{k-2}{k-1} i \right\rceil - 1}, \text{ for any integer } i$$

We describe the algorithm in a pseudo-ALGOL dialect including set operations ($\cup, \cap, -$) and list operations (first, last, "," for concatenation).

We first describe the procedure BEST(i,S) which is a tree selection algorithm used to determine the ordered list of the i best players of the set S. The set S is initially divided into two disjoint subsets S_1 and S_2, such that $S=S_1 \cup S_2$ and $|S_1| = 2^{\lceil \ell g |S| \rceil - 1}$. Furthermore, each set is associated with a list TOP(S) which is initially empty.

```
list procedure WINNER (list L1, list L2) := if last (L1) > last (L2)
then L1 else L2;
comment: WINNER uses one comparison except if one of the two
         lists is empty;
list procedure BEST (integer i, set S);
begin if S ≠ ∅ then
```

```
        begin for j = |TØP (S)|+1 until i do
              begin W:=WINNER (BEST (1, S1), BEST (1, S2));
                    if TØP(S1) = W then
                    begin TØP(S1):=Ø; S1:=S1-W; end;
                                  else
                    begin TOP(S2):=Ø; S2:=S2-W; end;
                    TØP(S):=TØP(S),W;
              end;
        end;
        TØP(S);
  end.
```

This tree selection algorithm performs at most $|S|-i+(i-1)\lceil lg|S|\rceil$ comparisons (see for instance [4] for further details). The new algorithm is an extension of this tree selection algorithm. Let P be the initial set of players which is divided into two disjoint subsets P_1 and P_2 such that $P_1 \cup P_2 = P$ and $|P_1| = 2^{\lceil lg|P|\rceil-1}$. The procedure BEST applied to P selects top players one by one in P_1 and P_2. The new algorithm uses two sequences of positive integers $\{u_\alpha\}$ and $\{v_\alpha\}$ and a characteristic step is to select either the u_h top players of P_1 or the v_j top players of P_2, according to the results of previous comparisons.

```
  list procedure SELECT (integer k, set P);
  begin h:=1; j:=1; A:=u1+v1;
        while A ≤ k do
  L1:   begin W:=WINNER (BEST(uh,P1), BEST (vj, P2));
              if TØP(P1) = W then
              begin TØP (P1):=Ø; P1:=P1-W;
                    h:=h+1; A:=A+uh;
```

```
            end;
                        else
            begin TØP (P2):=Ø; P2:=P2-W;
                  j:=j+1; A:=A+vj;
            end;
      end;
      R:=k - A + uh + vj;
      TØP(P):= TØP(P), PICK (BEST(R,P1), BEST(R,P2));
      Comment: TOP(P) contains the k best players of P, furthermore
               the k^th element of TOP(P) is the k^th player of P;
end.

list procedure PICK (list L1, list L2)
Comment: selects the top R players from the ordered lists
         L1 and L2 of length R using R comparisons;
Remark: It is possible (and sometimes more efficient) to
use the procedure SELECT recursively instead of the procedure
BEST. In that case, since the result of SELECT is not an ordered
list, it is also necessary to replace the procedure PICK.
```

Analysis of the Algorithm

An exhaustive analysis of the algorithm, to determine the best possible choices of $\{u_\alpha\}$ and $\{v_\alpha\}$ for given values of n and k, being quite tedious, we restrict our study to particular values of $\{u_\alpha\}$ and $\{v_\alpha\}$.

A comparison performed when line L1 of the algorithm is executed, or a comparison performed in the procedure PICK, clearly determines at least one new element of the pool of the k best players. Such a comparison will be referred to as an active comparison.

Case 1 $u_\alpha = v_\alpha = a$, $a \in N$, for all integer α.

Assuming that $k = ta$, $t \in N$, $a+t-1$ active comparisons are performed and clearly at most $n-2 + (k+a-2)$ $(\lceil \ell g n-2 \rceil)$ inactive ones. So that the difference between the number of comparisons performed by tree selection and the number of comparisons performed by this algorithm is clearly equal to:

$$k - \left[(a-1)(\lceil \ell g n \rceil - 1) + \frac{k}{a}\right].$$

The choice a=2 shows that this algorithm strictly improves on tree selection if $k > 2$ $(\lceil \ell g n \rceil - 1)$. In fact, there is an optimal manner of choosing a which is the closest integer to $\sqrt{\frac{k}{\lceil \ell g n \rceil - 1}}$.

For instance, suppose we are to select the 90^{th} player among a set of 2048. The choice $u_\alpha = v_\alpha = 3$, for all α, in our algorithm will save 39 comparisons over tree selection.

Case 2

We want to choose $\{u_\alpha\}$ and $\{v_\alpha\}$ such that, in the worst case, the number of inactive comparisons is equal to $n-2k + (k-1)\lceil \ell g n \rceil$. Such a choice guarantees that the algorithm is not worse than tree selection.

Assume that $n = 2^{i_1} + 2^{i_2}$ with $i_1 > i_2$. The values of $\{u_\alpha\}$ must satisfy the relation:

$$n-2-(1 - \sum_{i \le \alpha \le h} u_\alpha)(i_1 - 1) + (k-1- \sum_{i \le \alpha \le h} u_\alpha)(i_2 - 1) \le n-2k + (k-1)(i_1+1);$$

that is: $u_k \le 1 + \frac{i_1 - i_2}{i_1 - 1}(k-1- \sum_{1 \le \alpha \le h-1} u_\alpha)$.

The choice $v_\alpha = 1$ for all integer α appears to be always convenient, and a simple calculation yields that the algorithm improves strictly on tree selection if:

$$i_2 < \frac{(k-2)i_1 + 1}{k-1} .$$

For instance, for k=7, using the sequence $u_1=4$, $u_2=2$, $u_3=1$, saves 3 comparisons on tree selection if

$$2^{i_1} < n \leq 2^{i_1} + 2^{\left\lfloor \frac{u_1+1}{2} \right\rfloor}$$

For k=3, using $u_1=2$ and $u_2=1$ saves one comparison on tree selection if

$$2^{i_1} < n \leq 2^{i_1} + 2^{\left\lceil \frac{i_1+1}{2} \right\rceil} - 1 ,$$

and the new upper bound for $V_3(n)$ is

$$V_3(n) \leq n-3 + \lceil \lg(n-1) \rceil + \lceil \lg(n-2^{\left\lfloor \frac{\lg n}{2} \right\rfloor}) \rceil .$$

4. Acknowledgments

I am very grateful to C. L. Liu and J. A. Koch for their comments and suggestions during the preparation of this paper.

5. References

[1] Blum, M., R. Floyd, V. Pratt, R. Rivest, and R. Tarjan, "Linear time bounds for median computations", Proceedings of the Fourth Annual ACM Symposium on Theory of Computing, May, 1972.

[2] Caroll, L., St. Jame's Gazette, August 1, 1883, pp. 5-6.

[3] Hadian, A., and M. Sobel, "Selecting the t^{th} largest using binary errorless comparisons", Technical report 121, Department of Statistics, University of Minnesota, May, 1969.

[4] Knuth, D., "The Art of Computer Programming", Vol. 3, pp. 209-220, Addison-Wesley, 1973.

[5] Pratt, V. and F. Yao, "On lower bounds for computing the i^{th} largest element", Proceedings of the 14^{th} Symposium on Switching and Automata Theory, pp. 70-81, 1973.

[6] Yao, F., "On lower bounds for selection problems", Technical report MAC TR-121, Massachusetts Institute of Technology, 1974.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-74-651	2.	3. Recipient's Accession No.
4. Title and Subtitle BOUNDS FOR SELECTION			5. Report Date June 1974 6.
7. Author(s) Laurent Hyafil			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. GJ-41538
12. Sponsoring Organization Name and Address National Science Foundation Washington, DC			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

In this paper we show that the minimum number of comparisons necessary for the computation of the k^{th} element of a totally ordered set of size n, $V_k(n)$, is lower bounded by $n-k+(k-1)\lceil \log_2(\frac{n}{k-1})\rceil$. For $3 < k < \frac{n}{4}$, this bound improves the best lower bound presently known. A new algorithm which yields an upper bound that is better than the currently known bound for a large range of values of n will also be presented.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price

USCOMM-DC 40329-P71